



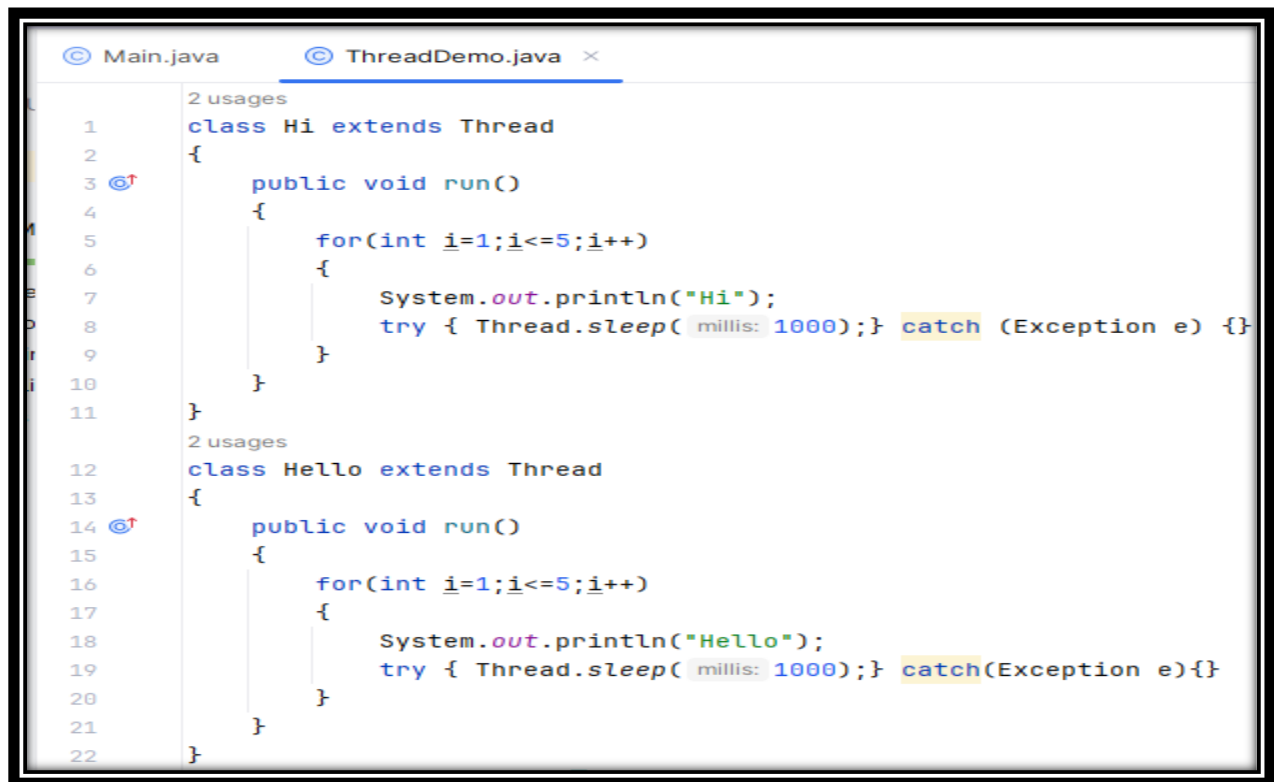
DEPARTMENT OF SOFTWARE ENGINEERING

Submitted By :Tanzeela Asghar
Submitted To :Sir Muhammad Shehzad
Roll no :2021-BSE-032
Course :Softwatre Construction Lab
Date: 22nd December 2023

Lab # 10
(Multithreading)

Multithreading:

Multithreading is a programming technique where multiple threads execute independently within a program. **Threads** are smaller units of a process, and they can run concurrently, providing benefits such as improved performance and responsiveness. In Java, multithreading can be implemented by extending the `Thread` class or implementing the `Runnable` interface. Synchronization mechanisms are used to ensure data consistency and avoid conflicts when multiple threads access shared resources. Despite its advantages, multithreading requires careful handling of issues like race conditions and deadlocks.



```
1 2 usages
2 class Hi extends Thread
3 {
4     public void run()
5     {
6         for(int i=1;i<=5;i++)
7         {
8             System.out.println("Hi");
9             try { Thread.sleep( millis: 1000);} catch (Exception e) {}
10        }
11    }
12 2 usages
13 class Hello extends Thread
14 {
15     public void run()
16     {
17         for(int i=1;i<=5;i++)
18         {
19             System.out.println("Hello");
20             try { Thread.sleep( millis: 1000);} catch(Exception e){}
21        }
22    }
```

```
public class ThreadDemo {  
    public static void main(String[] args)  
    {  
        Hi obj1 = new Hi();  
        Hello obj2 = new Hello();  
        obj1.start();  
        obj2.start();  
        try {  
            // Wait for both threads to finish before moving on  
            obj1.join();  
            obj2.join();  
        } catch (Exception e) {  
            System.out.println(e);  
        }  
  
        System.out.println("Threads finished.");  
    }  
}
```

```
C:\Users\92313\.jdk\openjdk-21\bin\java.exe "-javaagent:C:\Program Files\JetBrain  
Hi  
Hello  
Hello  
Hi  
Hi  
Hello  
Hi  
Hello  
Hello  
Hi  
Threads finished.  
  
Process finished with exit code 0
```

With Runnable Interface:

The `Runnable` interface in Java is used for creating threads. It allows a class to be executed by a thread without the need to extend the `Thread` class. Implementing `Runnable` involves defining a `run` method in the class. The class instance is then passed to a `Thread` object, which is started to execute the code concurrently. This approach promotes better code organization and resource sharing among threads.

```
© Main.java    © ThreadDemo.java ×
1 1 usage
2  class Hi implements Runnable
3  {
4      public void run()
5      {
6          for(int i=1;i<=5;i++)
7          {
8              System.out.println("Hi");
9              try { Thread.sleep( millis: 1000);} catch (Exception e) {}
10         }
11     }
12 1 usage
13  class Hello implements Runnable
14  {
15      public void run()
16      {
17          for(int i=1;i<=5;i++)
18          {
19              System.out.println("Hello");
20              try { Thread.sleep( millis: 1000);} catch (Exception e){}
21          }
22      }
23  }
```

```
23 > public class ThreadDemo {
24 >     public static void main(String[] args)
25     {
26         Runnable obj1 = new Hi();
27         Runnable obj2 = new Hello();
28         Thread t1 = new Thread (obj1);
29         Thread t2 = new Thread (obj2);
30         t1.start();
31         try { Thread.sleep( millis: 10);} catch (Exception e){}
32         t2.start();
33     }
34 }
35 |
```

Output:

```
C:\Users\92313\.jdk\openjdk-21\bin\java.exe "-javaagent:C:\Program Files\J
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello
Hi
Hello

Process finished with exit code 0
|
```

With Lamda expression:

A lambda expression in Java is a concise way to express anonymous functions. It provides a shorthand syntax for implementing functional interfaces. Lambda expressions are commonly used to write more readable and compact code, especially when working with functional programming features introduced in Java 8, like the Streams API. The basic syntax is `(parameters) -> expression`.

```
no usages
1  class Hi implements Runnable
2  {
3      public void run()
4      {
5          for(int i=1;i<=5;i++)
6          {
7              System.out.println("Hi");
8              try { Thread.sleep( millis: 1000);} catch (Exception e) {}
9          }
10     }
11     no usages
12     class Hello implements Runnable
13     {
14         public void run()
15         {
16             for(int i=1;i<=5;i++)
17             {
18                 System.out.println("Hello");
19                 try { Thread.sleep( millis: 1000);} catch (Exception e){}
20             }
21         }
22     }
```

```
public class ThreadDemo {
    public static void main(String[] args)
    {
        Thread t1 = new Thread (()->
        {
            for(int i=1;i<=5;i++){
                System.out.println("Hi");
                try { Thread.sleep( millis: 1000);} catch(Exception e){}
            }
        });
        Thread t2 = new Thread (()->
        {
            for(int i=1;i<=5;i++){
                System.out.println("Hello");
                try { Thread.sleep( millis: 1000);} catch(Exception e){}
            }
        });
        t1.start();
        try { Thread.sleep( millis: 10);} catch(Exception e){}
        t2.start();
    }
}
```

Output:

[illegible]